Gradus.jl: spacetime agnostic general relativistic ray-tracing through automatic differentiation

F. J. E. Baker,[1]★ and A. J. Young[1]
[1]H. H. Wills Physics Laboratory, Tyndall Avenue, Bristol BS8 1TL, UK

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT
We introduce Gradus.jl, an open-source...

Key words: keyword1 – keyword2 – keyword3

1 INTRODUCTION

In the era of quantitative, precision observational tests of General Relativity in the strong field regime it is necessary to have a fast and flexible method to compute the observational properties of accreting black hole systems. We have developed an open-source integrator Gradus.jl[1] for this purpose. In the remainder of the paper we describe how the software works, comparing with previous work in the literature, and outlining the new capabilities of Gradus.jl.

Transfer functions (**?**)

Julia is a high-performance... with SciML and DifferentialEquations.jl, a state-of-the-art ecosystem and workhorse for solving differential equations.

1.1 A short review of general relativistic ray tracing

- **review other softwares**
- **where does Gradus fit into the field (extensibility / maintainability / prototyping models)**

The trajectory of light in curved space may be determined by reformulating the Hamilton-Jacobi equations of motion as a first-order ordinary differential equation (ODE) system.

A second-order ODE system may alternatively be formulated directly from the geodesic equation; a method which is pedagogically simpler, but computationally more expensive than the first-order system, as either the full metric connection or derivatives of the metric must be explicitly implemented, else approximated at cost during runtime. With advancements in automatic differentiation, derivatives are cheap to compute, and consequently the second-order approach is tractable and both a parsimonious and spacetime agnostic method for computing geodesics.

2 NUMERICAL METHODS

For simplicity, we will here focus on static, axisymmetric spacetimes in the Boyer-Lindquist coordinates. Such a spacetime has a metric of the form

$$g_{\mu\nu} = g_{tt}\mathrm{d}t^2 + g_{rr}\mathrm{d}r^2 + g_{\theta\theta}\mathrm{d}\theta^2 + g_{\phi\phi}\mathrm{d}\phi^2 + 2g_{t\phi}\mathrm{d}t\mathrm{d}\phi. \tag{1}$$

We adopt $(-,+,+,+)$ signature and standard units $c = G = 1$. Greek indices (μ, ν) will be used to denote the four spacetime components, and Latin indices (i, j) for the three spatial components. We write partial derivatives with respect to the coordinates x^μ as $\partial_\mu := \partial/\partial x^\mu$.

2.1 Geodesic integration

The geodesic equation with coordinates x^μ is

$$\frac{\mathrm{d}^2 x^\mu}{\mathrm{d}\lambda^2} + \Gamma^\mu{}_{\nu\sigma} v^\nu v^\sigma = a^\mu, \tag{2}$$

where λ is the affine parameter and a^μ is some external acceleration. The effects of curvature on the trajectory encoded in the Christoffel symbols,

$$\Gamma^\mu{}_{\nu\sigma} := \frac{1}{2} g^{\mu\rho}\left(\partial_\nu g_{\rho\sigma} + \partial_\sigma g_{\rho\nu} - \partial_\rho g_{\sigma\nu}\right), \tag{3}$$

determined solely by metric and derivatives thereof. The geodesic equation is a set of four coupled second order differential equations that may be solved for a choice of initial x^μ and v^μ. A convention is to choose an initial position with $x^t = 0$, whereas the velocity vector is additionally constrained by the invariance

$$g_{\sigma\nu} v^\sigma v^\nu = \mu^2, \tag{4}$$

where μ is the invariant mass. This invariance gives rise to three solution classes depending on the sign of μ^2, namely $\mu^2 = 0$ corresponding to null-, $\mu^2 > 0$ to time-, and $\mu^2 < 0$ to space-like geodesics. Null geodesics are the trajectories of photons, time-like regular massive particles, and space-like geodesics are the trajectories of exotic particles, such as tachyons. Specifying the three-vector v^i determines v^t by Eq. (4), rearranged as

$$v^t = \frac{-g_{t\phi} v^\phi \pm \sqrt{-g_{ij} v^i v^j - \mu^2}}{g_{tt}}. \tag{5}$$

The choice of positive or negative root corresponds to the direction of time, wherein lies the ray-tracing *trick*: a time-reversal symmetry in the metric allows us to trace from an observer back to the point of

★ E-mail: fergus.baker@bristol.ac.uk (FB)
[1] Open-source and available under MIT license at https://github.com/astro-group-bristol/Gradus.jl.

origin, and then *reverse* time in order to calculate quantities as seen by the observer. **explain better**

The integration of the ODE system is performed numerically with an appropriate choice of algorithm. We favour the adaptive Tsitouras Runge-Kutta 5/4 (**?**). This integrator is shown to be fast and robust, also providing free fourth-order interpolants of the resulting geodesics. The interpolants can be particularly useful if additional quantities need to be re-traced along a geodesic, or for accurately finding points of intersections.

Derivatives of the metric needed in Eq. (3) may be efficiently computed with AD. This brings versatility and ease, as new spacetimes need only to define the metric for the geodesic system to be computable, and AD ensures the derivatives are free of the pathologies of other e.g. stenciling methods used to compute derivatives. If the class of spacetime exhibits additional symmetries, these can be exploited to reduce computation further: metrics of the form (1) can exploit $\partial_t g_{\mu\nu} = \partial_\phi g_{\mu\nu} = 0$ and avoid calculating two columns of the Jacobian entirely.

Sparsity is often symbolically inferable under simple operations. We use compile time Julia `@generated` functions, along with Symbolics.jl, to attempt to infer which terms in the Jacobian and Christoffel symbols can be avoided (if any), to generate optimal evaluation for a given class of metric.

2.2 Observers and emitters

To determine interpretable initial velocities it is useful to consider the coordinates local to the observer or emitter at coordinates x^μ. Following **?**, for observers one usually considers an image plane onto which a projection of geodesics is rendered, with geometry as in Fig. **??**. One may then define a set of impact parameters from components of the local momenta

$$\alpha := x^r \frac{p_{(\phi)}}{p_{(r)}}, \tag{6}$$

$$\beta := x^r \frac{p_{(\theta)}}{p_{(r)}}, \tag{7}$$

using indices in parentheses to denote the vector components in the local frame. Our choice of symbols (r, θ, ϕ) is to anticipate an identification between local and global bases, and to provide some intuition for the local frame.

Along with the invariance of four-momenta (c.f. Eq. (4)), one may obtain a curve of solutions for the local momenta

$$\frac{p_{(r)}}{p_{(t)}} = -\left(\sqrt{1 + \left(\frac{\alpha}{x^r}\right)^2 + \left(\frac{\beta}{x^r}\right)^2}\right)^{-1} = \mathscr{R}, \tag{8}$$

$$\frac{p_{(\theta)}}{p_{(t)}} = \mathscr{R}\frac{\beta}{x^r}, \tag{9}$$

$$\frac{p_{(\phi)}}{p_{(t)}} = \mathscr{R}\frac{\alpha}{x^r}. \tag{10}$$

For emitters, the above set of local initial momenta may be determined from a tangent vector pointing along the direction of a geodesic by decomposition (see also **??**). The component $p_{(t)}$ is the negative of the energy measured in the local frame, and is conventionally set to $-p_{(t)} = E = 1$ without loss of generality.

For both observers and emitters, we must identify a local frame, where the natural choice is the locally non-rotating frame (LNRF) (**?**) **check citation + explain significance**. The transformation from local to global coordinates is

$$p_\mu = \mathrm{e}^{(\nu)}{}_\mu \, p_{(\nu)} \tag{11}$$

where the basis vectors $\mathrm{e}^{(\nu)}{}_\mu$ are found using the theorem of Gram-Schmidt (, Appendix A). The formalism may be extended for an observer or emitter in motion, where their velocity modifies the mapping by a local Lorenz transformation, $\Lambda^{(\kappa)}{}_{(\nu)}$, as

$$p_\mu = \mathrm{e}^{(\nu)}{}_\mu \, \Lambda^{(\kappa)}{}_{(\nu)} p_{(\kappa)}. \tag{12}$$

2.3 Charts and horizons

A chart is used to terminate geodesic integration to avoid continuing computation when the fate of a given geodesic is determined. In practical terms, the chart is defined by a set of boundaries, and used to classify the outcome of an integration. As a motivating example, consider a chart with an inner and outer boundary: the inner boundary is a coordinate singularity of the metric, r_s (i.e. an event horizon), whereas the outer boundary is treated as the *effective infinity*, r_∞. Geodesics at the inner boundary are classified as lost behind the coordinate singularity, whereas those that reach the outer boundary are considered to escape to infinity with no further deviation to their trajectory. Additional boundaries of the chart may be used to represent accretion geometry: by terminating the integration when the geodesic crosses such a boundary, we consider the geodesic to have intersected the surface of the accretion disc.

The event horizon radius, r_s, used as the default inner boundary, may be calculated for a metric of the form (1) by solving

$$0 = \left.\frac{1}{g_{rr}}\right|_{x^r = r_s}. \tag{13}$$

For axisymmetric metrics, g_{rr} may be a function of both x^r and x^θ, in which case the inner boundary of the chart is a function of the poloidal coordinate. If no analytic function for r_s is known, it may be numerically approximated using root solving methods. We use root solvers from Roots.jl (). The default we use is their "Order 0" solver, a hybrid method that refines from secant to bracketing methods when possible.

In practice, close to the inner radius the adaptive time step of an ODE integrator tends to shrink dramatically due to near-singular derivatives, causing the integration to slow to almost a standstill. This may be avoided by scaling the inner horizon with the choice of constant $\mathcal{K} > 0$, such that $\tilde{r}_s = (1+\mathcal{K})r_s$, terminating the integration early when $x^r \leq \tilde{r}_s$. This constant may be adjusted depending on how vital it is for a geodesic to be able to glance the event horizon. We have chosen $\mathcal{K} = 10^{-2}$ by default, as it dramatically improves integration time without impacting the majority of simulations.

2.4 Computing observables

We consider observables to be any physical quantity calculated from a simulation that is evaluated using some or all of the points along a geodesic. Often only the start and end point of a geodesic are required to calculate some physical quantity, and under such circumstances it is computational beneficial to avoid allocating space for the full solutions.

A useful quantity to compute is the apparent redshift along a geodesic, due to both the Doppler and gravitational redshift. This may be compactly written as the ratio of energies

$$g = \frac{E_\text{end}}{E_\text{start}} = \frac{k_\mu u^\mu\big|_\text{end}}{k_\mu u^\mu\big|_\text{start}}, \tag{14}$$

where k^μ are the photon momenta, and u^μ the velocity of the emitting (start) and observing (end) media respectively.

2.5 Solving for special orbits

For simple disc models, including the α-disc of Shakura & Sunyaev (1973), the accreting matter is considered to follow Keplerian circular orbits in the equatorial plane, $x^\theta = \pi/2$, of the central singularity. These orbits are stationary points of the Hamiltonian and constrained by $v^r = v^\theta = 0$. For the class of static, axis-symmetric methods for particles with no external forces, they may be analytically determined (**?**, with a variation of their derivation in Appendix).

In other cases, such when the external acceleration $a^\mu \neq 0$ in (2), the analytic approach is often useful up to a point, before resolving to numerical methods for root finding.

Orbits may also be determined purely from the integration of geodesics, by mandating a stability measure and optimizing the initial velocity vector until the measure is a minimum. For example, let $\mathscr{M}$ measure the eccentricity of an orbit in the equatorial plane. For a given radius x^r, the velocity corresponding to a circular orbit is $v^\mu = v^t \partial_t + v^\phi \partial_\phi$. With (4), the velocity may be found through

$$\arg\min_{v^\phi} \ \mathscr{M}(x^r, v^\phi), \tag{15}$$

using a numerical optimizer. We use Nelder-Mead as the default algorithm. This optimization method is also used for arbitrary objective functions, such as for pinpointing geodesics that intersect chosen points ($\mathscr{M}$ measures closest approach) or exhibit specific desired features (e.g. $\mathscr{M}$ measures periodicity).

Circular orbits are classified as either stable or unstable, depending on the sign of $\mathrm{d}E/\mathrm{d}x^r$, with > 0 corresponding to stable configurations. Stable circular orbits are only possible for radii above the innermost stable circular orbit (ISCO) radius, $x^r \geq r_{\mathrm{ISCO}}$. The ISCO is the critical point at which

$$0 = \left.\frac{\mathrm{d}E}{\mathrm{d}x^r}\right|_{x^r = r_{\mathrm{ISCO}}}. \tag{16}$$

Within the ISCO is the so-called *plunging region* where $v^r \neq 0$. Circular orbits in this region are highly unstable and will either fall into the event horizon or escape to infinity if perturbed. These orbits may be numerically calculated from the ISCO four-velocity by offsetting $r_{\mathrm{ISCO}} - \delta x^r$, and integrating over a large λ interval. The components of v^μ may then be interpolated over x^r to approximate analytic solutions.

How we find circular orbits, how we find ISCO, photon radius, event horizon

2.6 Transfer functions

Both 1d (Cunningham) and 2d (lag-energy) transfer functions, the methods used to solve them, and the quadrature integration schemes.

2.7 Disc emissivity

We can calculate emissivity / flux maps for discs using either Voronoi tesselation or some symmetric prescriptions

2.8 Covariant radiative transfer

From (2), with $a^\mu = f^\mu/m$.

WIP? will probably finish implementing this before writing the full paper

3 DESCRIPTION OF THE CODE

Gradus.jl aims to have a single expressive high-level API for a variety of GRRT problems, with sensible defaults that work for most spacetimes. The code is accompanied by both code-level and generated documentation[2], with short tutorials and examples designed to provide a feature-rich overview and simultaneously teaching new users how to begin constructing their own simulations. The documentation strives to be the most up-to-date description of our numerical methods, detailing algorithm specific choices, and is rebuilt as part of our GitHub Workflows. The source code is written to be read by contributors and users alike to invite extension, and to be explicit about our methods, their benefits, and limitations.

The code makes use of Julia's heterogeneity and concurrency to run in multi-threaded and distributed environments, with GPU-offloading via DiffEqGPU.jl and CUDA.jl[3]. Gradus.jl is performant enough to create simulation products on personal computers[4], but scales effortlessly to supercomputers.

Precision is user-defined and inferred from user provided types. Single precision floating point arithmetic may be desirable for GPU computing, whereas 'big float' precision may be needed for some extreme near-horizon computations. In our discussion of the numerical methods, we referenced the current default algorithms used. Gradus.jl vendors many additional ODE solvers and numerical algorithms from the Julia SciML ecosystem, that may be used when certain integrations or problems require them.

Gradus.jl provides a number of predefined metrics, including the Kerr spacetime, Morris-Thorne wormhole, Johannsen-Psaltis, Dilaton-Axion, and a modified Kerr with coronal refraction. The Kerr-Newman metric is also implemented, complete with the ability to specify the electromagnetic potential vector, from which external forces in (2) are calculated. Furthermore, Gradus.jl allows for 1st order specification of the geodesic ODE system, primarily used as a point of comparison with the second order results.

For Julia, AD is implemented in the ForwardDiff.jl package (**?**)

Multi-threaded, multi-CPU, optionally GPU decelerated. Speedup depends on choice of solver (fixed time step faster on GPU)

List of currently implemented metrics

Adding new spacetimes

Accretion disc geometries

Point functions for composable results

Performance vs e.g. Bambi's NK and other work

3.1 Simulation products

What we can export, and how they can be exported / used in e.g. XSPEC

4 TEST PROBLEMS

4.1 Integration stability

Energy conservation, deflection problem, shadow, tests for naked-singularities

[2] `https://astro-group-bristol.github.io/Gradus.jl/dev/`

[3] We note that GPU acceleration is currently limited due to a combination of ODE callbacks and 'warp' termination. Progress in resolving outstanding issues is being tracked as an issue.

[4] E.g., recomputing all 600 `relline` transfer function tables on a 2021 Apple 8-core M1 Macbook takes approximately 2 hours.

4.2 Analytic solutions

All the special radii, the circular orbit energy

5 APPLICATIONS

5.1 Iron line profiles

Bambi's various metrics and relline, self consistency between methods

5.2 Lag-frequency spectra

Ingram's code? Jiachen's code

5.3 Emissivity curves

Wilkins and Fabian with lamp post and moving corona

6 CONCLUSIONS

Outlook

We encourage the community to contact us with interesting problems that may be tackled using Gradus.jlas we are happy to assist with new applications of the code.

ACKNOWLEDGEMENTS

This work is supported by the UKRI AIMLAC CDT funded by grant EP/S023992/1.

We thank Cosimo Bambi and Jiachen Jiang for sharing their software for testing purposes. FB thanks Rosie Baker for her debugging assistance, and Gabriella Kirkpatrick for her insights and conversations.

Budgetary codes and sponsors including AIMLAC, UKRI

DATA AVAILABILITY

REFERENCES

Shakura N. I., Sunyaev R. A., 1973, Astronomy and Astrophysics, 24, 337

APPENDIX A: ORTHOGONALIZATION AND LNRF WITH GRAM-SCHMIDT

How we derive the LNRF basis using Gram-Schmidt orthogonalization procedure, how we can use this to model corona

This paper has been typeset from a TEX/LATEX file prepared by the author.